Intelligent Neurosurgical Patient Profiling Model

Project Progress Report

Supervisor: Peter X.Liu

Team members:

Bingtao Liu 101029143

Xiang Cheng 101085518

Defa Hu 101038610

Jiayi Chen 101075620

Zejian Xing 101082830

January 22nd, 2022

Table of Contents

1.0 Introduction

The purpose of our project is to create models which are equivalent or better than the physician's current understanding of predicted particular outcomes. The project will involve training a neural network using CT/MRI scans of neurosurgical patients, which will essentially be series 2D black/white images.

In the first meeting, we divided the work of each team member and set the initial goals of the project. Over half of the academic year, as a team we had accomplished many goals that we had set out in the project proposal, some however still to be completed in the future. This project progress report will lay out the accomplishments and efforts associated, as well as issues that need to be addressed and future developments.

Other than what we have accomplished, we had a chance to discuss with our supervisor Professor Liu and came up with other functionalities. With plans on the expansion of our model up to 3 different models, along with data conversion built into the same packable python scripts, our script will be much more complicated than what we originally planned. The updated plan of our final product will include the development of the following scripts:

- MRI scan data conversion, converting.NII file into jpg with up to 100 slices of each scan, via Python scripts.
- Model 1, categorize images by the scanning direction, via machine learning
- Model 2, categorize tumors by the tumor type, via machine learning
- Model 3, categorize tumors by the tumor size (grading), via machine learning

Up till today, a complete machine learning framework had been coded in Python and tested, along with extensive data discovered and tested. We have successfully test trained many models with output accuracy up to 97%, however, we are unable to make use of such model to test our scans and images.

2.0 Theory and Techniques

This project, the Intelligent Neurosurgical Patient Profiling Model, is written in Python. The software part mainly includes machine learning, and file reading and conversion.

2.1 Machine learning theories and techniques

At first, we learned how to use TensorFlow in Machine learning. One important point is the tensorflow.keras, which is the API used to build and train models.
(https://www.tensorflow.org/tutorials/keras/classification)

From the TensorFlow website [1], we get the method to store the training data, for example, save the training data into two arrays such as train_images and train_labels. The model will learn the data related to the images and the labels. The next step is to make the predictions about the test dataset, after that, verify the predictions which match the labels.

The most important part of training the model is using “model.fit”.
We use the code as ***model.fit(“our training images dataset”, “our training labels”, “the epochs = the train times”)***

In the code:
import tensorflow
This is to import TensorFlow’s library into our code for further use.

from tensorflow.keras.preprocessing import image
Importing the image class from TensorFlow library, for importing further classes.

from tensorflow.keras.models import Sequential
Sequential is a method to group a linear stack of layers into the model, appropriate for the layers with one input tensor and one output tensor.

from tensorflow.keras.layers import Conv2D

Conv2D obtained from layers class, used for achieving higher accuracy, is usually used for images rather than videos.

from tensorflow.keras.layers import MaxPooling2D

MaxingPooling2D is one of the Max pooling operations, and is specific for 2D spatial data. Max pooling operations is to calculate the maximum values of the images.

from tensorflow.keras.layers import Flatten

Flatten is the function available in the TensorFlow library and reduces the input data into a single dimension instead of 2 dimensions and it will not affect the batch size. Flatten was used in our project for further processing of data.

from tensorflow.keras.layers import Dense

This function is used to create fully connected layers, in which every output depends on every input for further machine learning.

from tensorflow.keras.preprocessing.image import ImageDataGenerator

ImageDataGenerator is to generate batches of tensor image data with real-time data augmentation.

import os

Python OS module provides easy functions that allow the user to interact and get Operating system information irrespective of it being a Windows Platform, Macintosh or Linux. For our project, Windows Platform was used to process files from different places in the system.

import numpy as np

Numpy stands for Numerical Python, it is a scientific computing library built on top of the Python programming language. The as np portion of the code then tells Python to give NumPy the alias of np and allow the user to use the NumPy function built in it to quickly create and analyze data.

import pandas as pd

To import the library pandas to Python. Pandas is a library of Python used for data manipulation and analysis.

from tensorflow.keras.models import model_from_json

JSON is a simple file format for describing data hierarchically. Keras provides the ability to describe any model using JSON format with a to_json() function. This can be saved to a file and later loaded via the model_from_json() function that will create a new model from the JSON specification.

For the dataset of our project, we input the data files in the read-only files directory.
The code we used as below:

```
for dirname, _, filenames in os.walk('The input files of our dataset'):
    for file in filenames:
        print(os.path.join(dirname, file))
```

For the Convolution neural network setting [5], we used the code below:

```
classifier = Sequential()

classifier.add(Conv2D(filters = 32,kernel_size = (3, 3),
                      input_shape = (512, 512, 3), activation =      "relu"))

classifier.add(MaxPooling2D(pool_size = (6,6)))

classifier.add(Conv2D(filters = 32,kernel_size = (3, 3), activation = "relu"))
classifier.add(MaxPooling2D(pool_size = (6,6)))

classifier.add(Flatten())

classifier.add(Dense(units = 128, activation = "relu"))
classifier.add(Dense(units = 4, activation = "sigmoid"))

classifier.compile(optimizer = "adam", loss = "CategoricalCrossentropy", metrics =
["accuracy"])
```

Firstly, we created a network by using Sequential(). Then we started to add layers. For the layers, we choose to use 32 filters which means the model will focus on 32 features, and

kernel_size = (3,3), this is the size of the filter. The size of input images is (512,512). After the layers are set up, we have one more step to do before starting training. we need to compile the model. In the method compile(), loss is loss function, which measures how accurately the model is during training; optimizer sets how the model is updated based on the data it sees and its loss function; metrics is to monitor the training and testing steps.

After the layers are set up properly, we need to think about how to load image data to the model. For loading the data we use the following code:

```
train_datagen = ImageDataGenerator(
                                   rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)
```

```
training_dataset = train_datagen.flow_from_directory('data/Training',
                                                     target_size=(512, 512),
                                                     batch_size=32,
                                                     class_mode='categorical')

testing_dataset = test_datagen.flow_from_directory('data/Testing',
                                                   target_size=(512, 512),
                                                   batch_size=32,
                                                   class_mode='categorical')
```

In our model, we use ImageDataGenerator, this class provides a quick and easy way to augment the images. The reason why we need to augment the images is that it makes the model more accurate [2]. In this case, we rescale the image, set the sheer intensity, the range for random zoom, and allow the images to flip randomly horizontally. Then we use flow_from_directory() to load the dataset from path ‘data/Training’ for the training dataset, and ‘data/Testing’ for testing dataset. This dataset contains the data images and their labels. In our program, we trained our model as the bellowing code:

```
    classifier.fit(training_dataset,
                   steps_per_epoch = 44,
                   epochs = 100,
                   validation_data = “our testing dataset”,
                   validation_steps = 8)
```

In this function, the “steps_per_epoch = 44” means every single image will be learned 44 times in the training process. And the “epochs = 100” explains the times that all of the images are trained. The validation function will exhibit the loss rate and the accuracy rate of each epoch when the training is in progress. Validation_steps indicates that the whole validation will take place however many times with a single batch of testing data.

To save our model for later use. One step is to serialize our model to JSON and another step is to serialize the weights to HDF5. The HDF5 is a basic save format which is provided by Keras. The code of saving model is shown below:

```
from keras.models import model_from_json
model_json = classifier.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
```

```
classifier.save_weights("model.h5")
print("Saved model to disk")
```

And to load the JSON and load the model we already trained:

```
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
```

```
loaded_model.load_weights("model.h5")
print("Loaded model from disk")
```

After the model is trained and saved, we need individual predictions for our trained model which is the test of the model.

The code we used to test model as below:

```
test_image = image.load_img('Our test dataset files', target_size = (512, 512))

test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
```

```
result = classifier.predict(test_image) # Prediccion
print(training_dataset.class_indices)
print(result)
```

And we can also evaluate the accuracy on the test data, we can use the below code:

```
loss, acc = classifier.evaluate(testing_dataset,verbose=2)
print('Restored model, accuracy: {:5.2f}%'.format(100 * acc))
```

Since we have already compiled the model, we don't need to do it again here. We use evaluate() and testing_dataset as the input data to check the accuracy of the model.

2.2 Converting .nii file to .jpg

In the process of development, we found that it is easier and faster to train the model using .jpg image than using .nii image. However, it is not easy to find the specific image we expect from the internet. We found that we can convert one .nii image to many .jpg images, this is the code we used for converting [4]:

```
def read_niifile(niifile):
    img = nib.load(niifile)
    img_fdata = img.get_fdata()
    img90 = np.rot90(img_fdata)

    return img90

def save_fig(file):
    fdata = read_niifile(file)
    (y,x,z) = fdata.shape
    for k in range(z):
        silce = fdata[:,:, k]
        imageio.imwrite(os.path.join(output, '{}.jpg'.format(os.path.splitext(i)[0][0]+'_'+str(k))),silce)

def findAllFile(base):
    for root, ds, fs, in os.walk(base):
        for f in fs:
            yield f

base = r'../'
output = r'../img'
for i in findAllFile(base):
    dir = os.path.join(base, i)
    savepicdir = (os.path.join(output,i))
    save_fig(dir)


save_fig("data1.nii")
```

3.0 Results and Discussion

As stated, we have successfully trained models with accuracy up to 97% and tested with existing testing data, where the results are highly accurate. The progress output for the training process is shown below:

This is the dataset we found online [3]:

First, the data are separated into two files, Testing, and Training. Then in both folders, we have four more files that contain the specific tumor image. The name of the files will be the labels of the dataset.

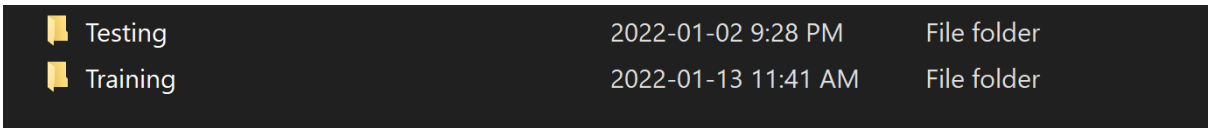


Figure 1. Testing and Training Data Files

Figure 2. Four Different Tumors Data Files

The image below is a screenshot of a portion of our no-tumor testing data.

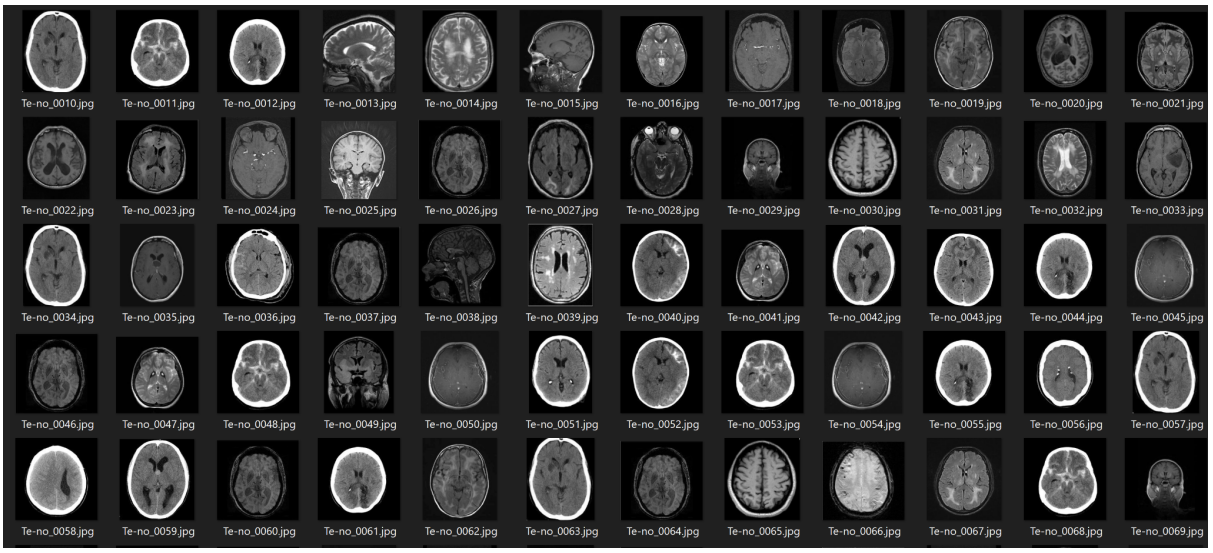


Figure 3. MRI Scan Images in Testing Data File

Training (section)

```
# Loading data into machine learning function
Found 5712 images belonging to 4 classes.
```

Found 1311 images belonging to 4 classes.

.....

######################

The Epoch10/100 stands for the epoch evaluation processed.

The 20/44 [=============>.....] stands for 20 steps out of all 44 batch data has been evaluated and trained.

Loss stands for when training the some data had been lost due to adjusting weights, and the accuracy stands for the fitting accuracy of such model.

######################

Epoch 10/100

44/44 [==============================] - 219s 5s/step - loss: 0.7661 - accuracy: 0.6814

Epoch 11/100

44/44 [==============================] - 229s 5s/step - loss: 0.7557 - accuracy: 0.6907

Epoch 12/100

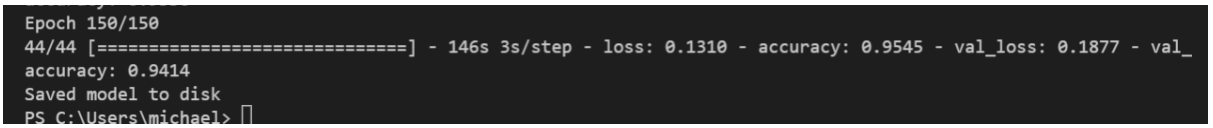
44/44 [==============================] - 229s 5s/step - loss: 0.7272 - accuracy: 0.7063

Epoch 13/100

20/44 [=============>.....] - ETA: 1:56 - loss: 0.7264 - accuracy: 0.7078

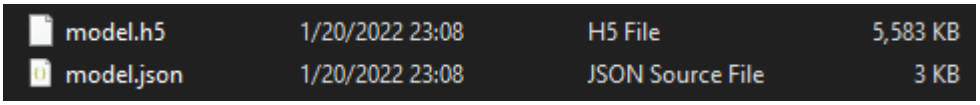
.....

We took one screenshot of the training progress. The training process is the most time-consuming of our entire program. In our project, we used 5712 images to train our model, it usually took 7-8 hours to finish the training task.



After the model had been saved the script prints text indicating the model had been saved.

Saved model to disk



The saved trained model is, as shown above, stored as HDF5 and JSON files.

The next section of our project is to load the saved model and obtain the test results. Here is one example showing a glioma tumor, the patient image is as shown below.

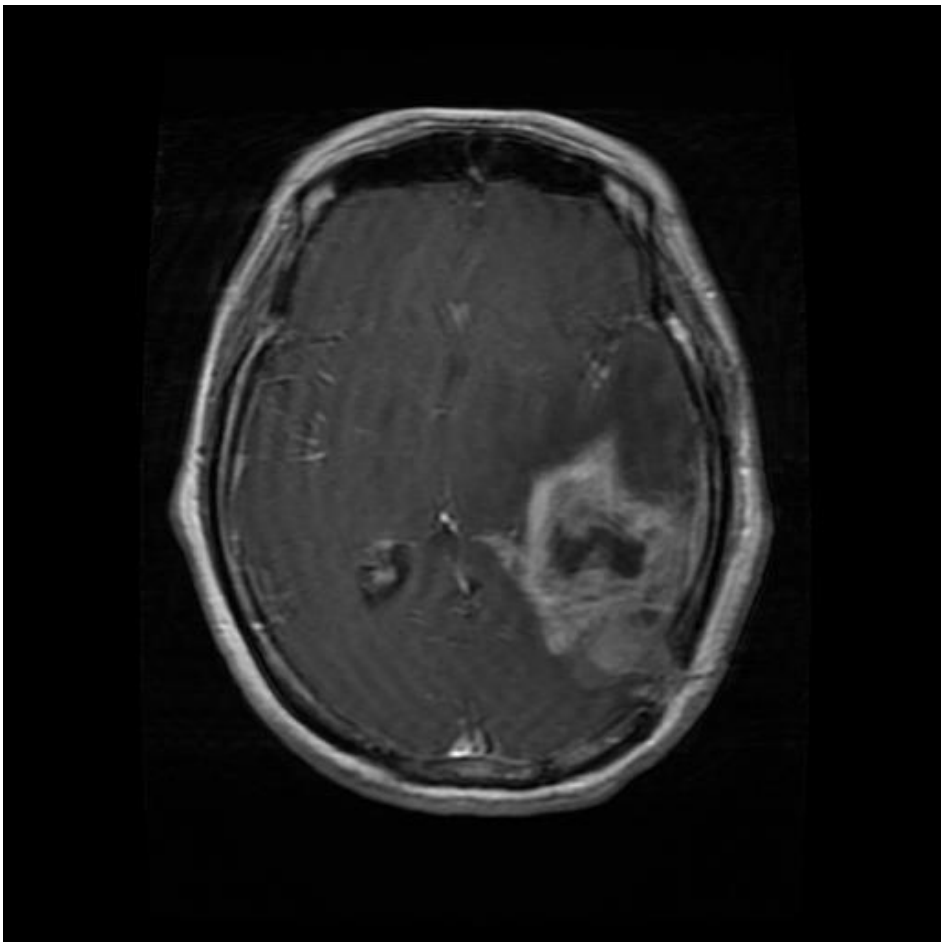


Figure 4. MRI Scan Images with Glioma Tumor

Below is the result obtained from our test terminal.

```
# Loading model
Loaded model from disk
# Printing the indices of trained model
{'glioma': 0, 'meningioma': 1, 'notumor': 2, 'pituitary': 3}
# Predicting matching likelihood of each type of tumor
[[0.7069714  0.28204957 0.65776855 0.59941036]]
# Compare the matching percentage and output index corresponding to the highest number
0
```

Following that, we did another test with another patient's MRI image with a glioma tumor. We took a screenshot of the result in our terminal to display this progress.

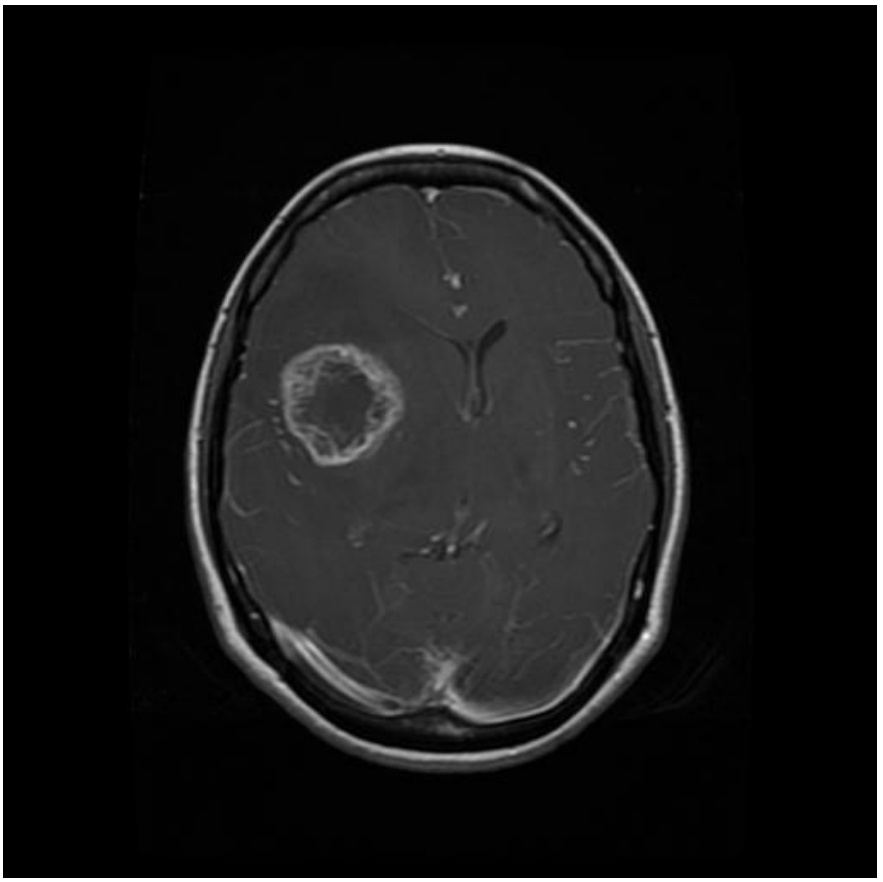


Figure 5. Another MRI Scan Images with Glioma Tumor

```
107    4th year project/Testing/notumor/Te-no_0045.jpg', target_size = (512, 512)) # Cargamos la imagen
108
109
110

PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE

raries. Please make sure the missing libraries mentioned above are installed properly if you would like to use
GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required libr
aries for your platform.
Skipping registering GPU devices...
2022-01-05 10:51:31.292527: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is opti
mized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-cri
tical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Found 5712 images belonging to 4 classes.
Found 1311 images belonging to 4 classes.
Loaded model from disk
{'glioma': 0, 'meningioma': 1, 'notumor': 2, 'pituitary': 3}
[[0.65822595 0.35966218 0.32357174 0.10591862]]
0
```

Figure 6. Testing Result with Glioma Tumor

As the result displays, there are 5712 MRI images in our training files and 1311 images in our testing files. The result indicates the probability that the image is shown here is one of these three kinds of tumor cases or no tumor case.

From the testing results, the probability of glioma tumor is around 66%, which is much higher than the other three situations. We took this result as one successful task.

In our next test, we used one image which contains another case of tumor type, meningioma tumor. Below is the MRI image and the test result screenshot. As the result shows, the probability of a meningioma tumor for this patient's tumor is around 71%. We consider this as another successful task as well.

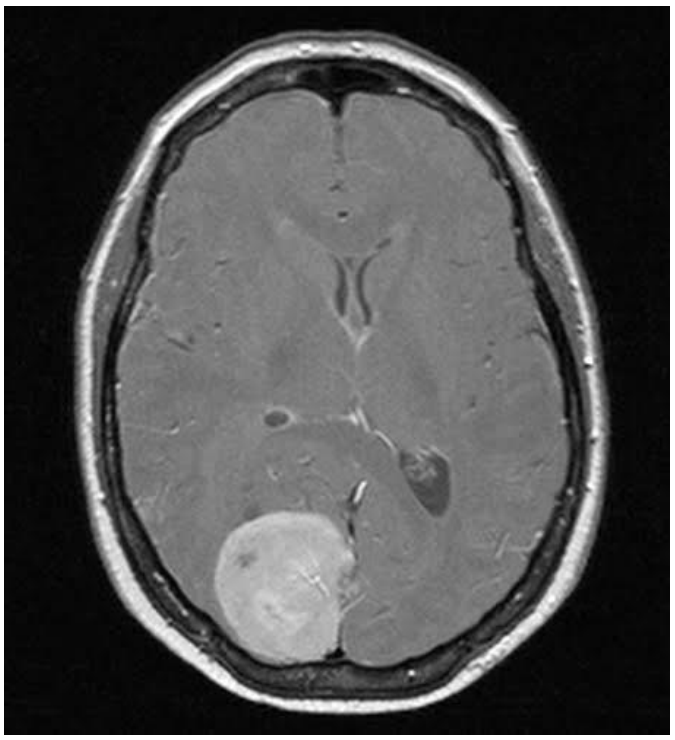


Figure 7. MRI Scan Images with Meningioma Tumor

```
107   C 4907 4th year project/Testing/glioma/Te-gl_0058.jpg', target_size = (512, 512)) # Cargamos la im
108   ores
109
110

PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE

raries. Please make sure the missing libraries mentioned above are installed properly if you would like to use
GPU. Follow the guide at https://www.tensorflow.org/install/gpu for how to download and setup the required libr
aries for your platform.
Skipping registering GPU devices...
2022-01-05 09:33:47.552692: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is opti
mized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-cri
tical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Found 5712 images belonging to 4 classes.
Found 1311 images belonging to 4 classes.
Loaded model from disk
{'glioma': 0, 'meningioma': 1, 'notumor': 2, 'pituitary': 3}
[[0.30147174 0.708658   0.37964556 0.51311463]]
1
PS C:\Users\Jeff\OneDrive - Carleton University\2021-2022 Carleton U\Fall\SYSC 4907 4th year project>
```

Figure 8. Testing Result with Meningioma Tumor

After testing the MRI images with the tumor, we tested another MRI image without the tumor. We selected one image in our no tumor image files. To get higher accuracy, we modified our code to hyperparameters.

We modified

```
classifier.add(Dense(units = 4, activation = "sigmoid"))
```

to

```
classifier.add(Dense(units = 4, activation = "softmax"))
```

And we get the result as below showing:

```
{'glioma': 0, 'meningioma': 1, 'notumor': 2, 'pituitary': 3}
[[0. 0. 1. 0.]]
2
```

The result obtained was correct. There are [0.0.1.0] displayed in our result which means the program considered our image as the no tumor case. And the changing of sigmoid and softmax also exhibit the difference in the result. There is not any probability of being accurate to decimal, instead, the “0” indicates the patient is free of this category, the “1” indicates the patient is in this category.

4.0 Further Goal

In this project, the basic goal in the future is to promote and popularize the auxiliary diagnostic software to the health industry to reduce the time waste caused by the diagnosis and even serve as auxiliary teaching for novice medical staff.

To achieve this, we need to figure out a few more things:

The MRI file conversion is the first part to be done, but the speed of the conversion can be increased again to reduce the overall completion time.

Model 1, the diagnosis of tumors, has been initially completed, but for the completion of this project, we need to improve the accuracy of the model. So it takes extra time to continue collecting files and deeply training the model.

The rough design of model 2 has been completed. This step will reclassify the tumor results in model 1. The results will have the three most widespread tumors: glioma tumor, meningioma tumor, and pituitary tumor. However, the implementation of Model 2 is not so accurate, and the code needs to be improved. Subsequent implementations will perform the function of classifying tumors while maintaining high accuracy.

Model 3 is the most difficult of all models, and more data is needed to support our research on this model. At present, the project is halfway through, and part of the design and implementation has not been fully completed. Subsequent design and implementation will complete the tumor size determination.

Finally, because our projects are done separately. After we have all the models, we need to write a separate file before they can be combined and used.

5.0 Contributions

Bingtao Liu:

Completion of hyperparameter optimization.

Models update and testing. Convolution layers setting up.

The conception of construction and training in Model 1.

Completion of Model 3 designing and training.

Xiang Cheng:

Model 2 categorizes tumors by tumor type - refinement, updates, and testing.

Investigated different machine learning options.

Defa Hu:

Completion of the code to convert .NII file (MRI scans) into jpg with up to 100 slices of each scan, via Python scripts.

Design and implement the save/load and evaluation part of the project.

Jiayi Chen:

Collect required brain MRI scans (Brain MRI with no tumor, glioma tumor, meningioma tumor, and pituitary tumor) from Google's Open Image Database.

Participate in the conception and completion of Model 1.

Participate in the conception of Model 3.

Zejian Xing:

Library collecting, library importing in Model 2.

Train/test set defining, evaluating, and testing

6.0 Timetable

Date	Task	Deliverable
October 22, 2021	Project proposal	Project proposal
November 5, 2021	Images dataset collection	
November 19, 2021	Library collection, image processing	Image dataset
December 3, 2021	Draft Model build	Draft Model
December 10, 2021	Model train	
December 31, 2021	Hyper parameter	
January 5, 2022	Model test, Hyper parameter, Model update	Prototype Models
January 15, 2022	Working on Progress report	
January 21, 2022	Models update and correction	
January 22, 2022		Progress report
January 28, 2022	Project update	
February 11, 2022	Test and Debug	Updated project models
February 25, 2022	Oral Presentation preparation	Final Report Draft

March 11, 2022	Working on Final Project Report	Oral Presentation
April 12, 2022		Final Project Report

7.0 Conclusion

In conclusion, as we continue to move on along the academic year, we believe that we have met our expectations and made progress ahead of our schedule, including machine learning shell and image conversion. We are on the right track in creating the model.

However, there are still plenty of issues and improvements that need to be addressed as we move along. We still need to set recurring meetings twice a week to achieve the features and improvements described in the Future Goals section. It is believed that with more effort and time, we will be able to manage and achieve the goals that we have set out in the proposal.

8.0 Reference

[1] TensorFlow. Image classification [Online]. Available: https://www.tensorflow.org/tutorials/images/classification#train_the_model [Accessed: 1-Oct-2021].

[2] Bhandari, A. (2020, August 16). *Image Augmentation Keras | Keras ImageDataGenerator*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/

[3] SartajBhuvaji, "Sartajbhuvaji/brain-tumor-classification-dataset," *GitHub*, 24-May-2020. [Online]. Available: https://github.com/sartajbhuvaji/brain-tumor-classification-dataset. [Accessed: 10-Jan-2022].

[4] 批量*nii*文件转换为*png*图像. 批量nii文件转换为png图像_Begining-CSDN博客_nii转png. (n.d.). Retrieved January 21, 2022, from https://blog.csdn.net/weixin_43330946/article/details/89576759

[5] TensorFlow. Image classification [Online]. Available: https://www.tensorflow.org/tutorials/images/cnn [Accessed: 1-Dec-2021].